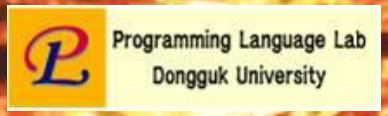
객체지향 언어와 실습

실습 8주차

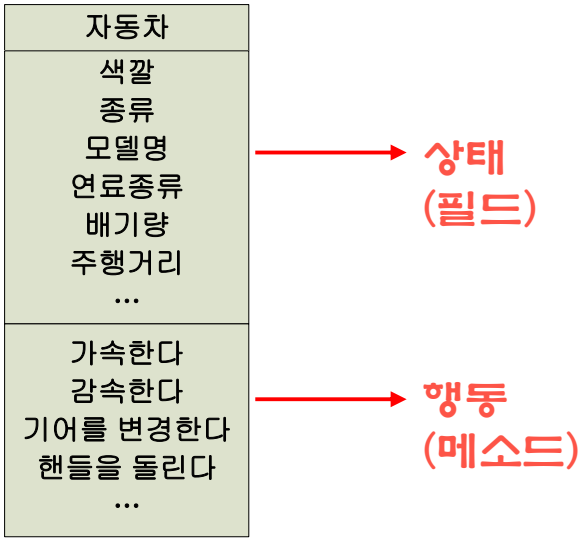


목차

- 클래스와 객체
- 영화 클래스
- 성적을 처리하는 프로그램
- 분수 클래스
- 실습 과제

클래스와 객체

- 클래스(Class)
  - 자바 프로그램의 기본 집합
  - 객체의 상태와 행동을 표현하는 방식
  - 객체를 생성하기 위한 틀
- 객체(Object)
  - 현실 세계에 존재하는 특정 대상의 표현
  - 클래스를 사용하여 특정 대상으로 실체화



영화 클래스 [1/5]

- 클래스 정의
  - 필드 내용: 제목, 장르, 상영시간(분)
  - 생성자 필드 값 입력
  - 메소드: play() <"영화제목"을(를) "상영시간" 분간 상영합니다.> 출력

```
 1 class Movie {
 2       public String title;
 3       public String genre;
 4       public int runningTime;
 5       Movie() { };
 6       Movie(String title, String genre, int runningTime) {
 7             this.title = title;
 8             this.genre = genre;
 9             this.runningTime = runningTime;
10       }
11       public void play() {
12             System.out.println(title + "을(를) " + runningTime + "분간 상영합니다.");
13       }
14 }
```

영화 클래스 [2/5]

- 테스트 클래스

```
1  public class MovieTestDrive {
2        public static void main(String[] args) {
3              Movie m1 = new Movie();
4              m1.title = "영화1";
5              m1.genre = "공포";
6              m1.runningTime = 90;
7
8              Movie m2 = new Movie("영화2", "액션", 120);
9
10             m1.play();
11             m2.play();
12
13       }
14 }
```

영화 클래스 [3/5]

- 필드의 접근수정자를 private로 변경

```
1  class Movie {
2        private String title;
3        private String genre;
4        private int runningTime;
5        Movie() { };
6        Movie(String title, String genre, int runningTime) {
7              this.title = title;
8              this.genre = genre;
9              this.runningTime = runningTime;
10       }
11       public void play() {
12             System.out.println(title + "을(를) " + runningTime + "분간 상영합니다.");
13       }
```

영화 클래스 [4/5]

- 영화 클래스 getter/setter 메소드 추가

```
 1 public void setTitle(String title) {
 2       ...
 3 }
 4 public void setGenre(String genre) {
 5       ...
 6 }
 7 public void setRunningTime(int runningTime) {
 8       ...
 9 }
10 public String getTile() {
11       ...
12 }
13 public String getGenre() {
14       ...
15 }
16 public int runningTime() {
17       ...
18 }
```

영화 클래스 [5/5]

- 테스트 클래스 수정

```
 1 public class MovieTestDrive {
 2       public static void main(String[] args) {
 3             Movie m1 = new Movie();
 4             m1.setTitle("영화1");
 5             m1.setGenre("공포");
 6             m1.setRunningTime(90);
 7
 8             Movie m2 = new Movie("영화2", "액션", 120);
 9
10             m1.play();
11             m2.play();
12
13       }
14 }
```

성적을 처리하는 프로그램 [1/4]

- 성적을 처리하는 사람의 관점에서 생각해 보자
  - 성적은 학생 단위로 처리, 수업은 국,영,수 3과목
  - 학생에 대한 정보
    - 이름, 과목별 점수, 총점, 평균, 학점
    - 처리 중인 학생수
  - 생성자
    - 이름만 입력하는 생성자, 이름+모든 시험점수까지 입력하는 생성자
  - 메소드
    - 모든 과목의 시험 점수입력 메소드
    - 총합점수, 평균점수 반환 메소드
    - 평균점수가 91이상 A, 81~90은 B, 71~80은 C, 61~70는 D, 그 이하에선 F 학점 반환 메소드
    - 모든 학생의 정보를 출력하는 메소드

성적을 처리하는 프로그램 [2/4]

- 필드와 생성자

```
 1 class Student {
 2       private String name;
 3       private int kor;
 4       private int eng;
 5       private int math;
 6       private static int cntCurrentStudent = 0;
 7
 8       public Student(String name) {
 9             this.name = name;
10             kor = eng = math = 0;
11             cntCurrentStudent++;
12             System.out.println(name + " 학생이 생성되었습니다.");
13             System.out.println("현재까지 처리중인 학생수는 " + cntCurrentStudent + "명입니다.");
14       }
15       public Student(String name, int korean, int english, int mathematics) {
16             this(name);
17             setScore(korean, english, mathematics);
18       }
```

성적을 처리하는 프로그램 [3/4]

- 메소드

```
19      public void setScore(int korean, int english, int mathematics) {
20            kor = korean;
21            eng = english;
22            math = mathematics;
23      }
24      public int total() {
25            return kor + eng + math;
26      }
27      public int average() {
28            return total()/3;
29      }
30      public char grade() {
31            if(average() > 90)        return 'A';
32            else if(average() > 80)   return 'B';
33            else if(average() > 70)   return 'C';
34            else if(average() > 60)   return 'D';
35            else                      return 'F';
36      }
37      public void printStudentInfo() {
38            System.out.println(name + " 학생의 성적 정보는 다음과 같습니다.");
39            System.out.println("국어 : " + kor + ", 영어 : " + eng + ", 수학 : " + math);
40            System.out.println("총점 : " + total() + ", 평균 : " + average() + ", 학점 : " + grade());
41      }
42  }
```

성적을 처리하는 프로그램 [4/4]

- 성적 처리

```
 1 public class Ex01 {
 2       public static void main(String[] args) {
 3             Student s1 = new Student("홍길동");
 4             s1.setScore(75, 88, 54);
 5             Student s2 = new Student("김영희", 90, 100, 94);
 6             Student s3 = new Student("김철수", 12, 54, 38);
 7
 8             s1.printStudentInfo();
 9             s2.printStudentInfo();
10             s3.printStudentInfo();
11       }
12 }
```

Problems | Javadoc | Declaration | Console

<terminated> Ex01 [Java Application] C:₩Program Files₩Java

```
홍길동 학생이 생성되었습니다.
현재까지 처리중인 학생수는 1명입니다.
김영희 학생이 생성되었습니다.
현재까지 처리중인 학생수는 2명입니다.
김철수 학생이 생성되었습니다.
현재까지 처리중인 학생수는 3명입니다.
홍길동 학생의 성적 정보는 다음과 같습니다.
국어 : 75, 영어 : 88, 수학 : 54
총점 : 217, 평균 : 72, 학점 : C
김영희 학생의 성적 정보는 다음과 같습니다.
국어 : 90, 영어 : 100, 수학 : 94
총점 : 284, 평균 : 94, 학점 : A
김철수 학생의 성적 정보는 다음과 같습니다.
국어 : 12, 영어 : 54, 수학 : 38
총점 : 104, 평균 : 34, 학점 : F
```

분수 클래스

- 분수(Fraction) 클래스를 구현해보자.
  - 필드 : 분자(numerator), 분모(denominator) ➔ private 접근수정자
  - 생성자 : 분자, 분모 값을 입력 받아 저장
  - 메소드
    - add(Fraction operand) 피연산 분수를 입력 받아 더한다.
    - mul(Fraction operand) 피연산 분수를 입력 받아 곱한다.
    - toString() 분수를 "분자/분모" 형태의 스트링으로 반환한다.
- 테스트 클래스에서 분수클래스를 활용해 연산결과를 확인해본다.

실습 과제

- 분수클래스 완성
  - 빼기와 나누기 메소드를 구현한다.
  - 기약분수 변환 메소드를 구현한다.
    - 기약분수:분모와 분자가 1이외의 공통된 인수를 갖지 않는 분수
    - 기약분수변환 메소드를 이용해 이미 구현된 연산 메소드들의 결과가 기약분수로 저장되도록 변경